A Memory Efficient Grapheme-to-Phoneme Conversion System for Speech Processing

Jun Huang, Lex Olorenshaw, Gustavo Hernández-Ábrego, Lei Duan

Spoken Language Technology Group
Sony Electronics, USA
jinh@slt.sel.sony.com, lex.olorenshaw@am.sony.com, gustavo@slt.sel.sony.com, lei@slt.sel.sony.com

Abstract
In this paper, a memory efficient, statistical data driven approach is proposed and successfully tested for grapheme-to-phoneme (G2P) conversion. In our system, a dynamic programming (DP) based first algorithm is formulated to estimate the optimal joint segmentation between training sequences of graphemes and phonemes. A statistical language model is trained to model the contextual information between grapheme and phoneme segments. A two-stage fast decoding algorithm is also proposed to recognize the most likely phoneme sequences given the input test word and the n-gram grapheme models. Experimental results show that this system has similar recognition accuracy as a decision-tree based g2p system and requires much less memory and processing time.

1. Introduction
During the past decades, many different methods have been proposed for g2p conversion for the applications of dynamic speech recognition and text-to-speech (TTS) synthesis. These methods can be classified into two broad categories, namely the deterministic rule-based approach and the data-driven approach. In an entirely rule-based approach, the g2p rules are generated by phonetic/linguistic experts to find an exhaustive list of the mappings between graphemes and phonemes. This approach is very time consuming, language dependent and not very accurate for some languages such as English. (One classic entirely rule-based g2p system carried out by researchers in the Naval Research Lab can be downloaded from [7].) The data-driven approaches can be further classified into three different sub-categories, namely inductive learning (IL), pronunciation by analogy (PBA), and grapheme model.

In the IL approach, some statistical learning methods such as decision trees [5] were used to learn the g2p rules. The IL systems can generate reasonable phoneme accuracy (> 90%), but they usually require a lot of system memory (in the order of tens of megabytes in our experiments). The PBA systems adopt a more global consideration of the context information within grapheme sequences [1] in the process of g2p. However, this method also consumes a lot of system memory (again in the order of tens of megabytes in our experiments) in order to build the pronunciation neighborhood for the entire training dictionary. Furthermore, the phoneme error rate is higher than that of the decision-tree based methods. A grapheme is a pair composed by a letter sequence and a phoneme sequence of possibly different lengths [2,3]. In a grapheme-based approach, some statistical modeling technique is used to learn the optimal alignment between the sequences of graphemes and phonemes, and the joint probabilities of each grapheme.

Two examples of grapheme-based g2p conversion systems were proposed in [2,6]. Our system differs from previous approaches in the following three major aspects: 1) our g2p system aims to provide more complete lexical information which includes not only the pronunciation, but also some prosodic information such as stress markers. 2) the major objective of our project is to design a multi-lingual g2p system with efficient memory consumption for potential application in electronic devices with limited system memory and processor speed. As a result, an efficient two-dimensional DP-based alignment algorithm is proposed to estimate the optimal joint-segmentation between the training word orthographies and their corresponding pronunciations, instead of using HMMs to obtain alignments as described in [6]. 3) In the decoding stage, a fast and memory-efficient, two-stage decoding algorithm is proposed in our system for g2p conversion, instead of directly using the standard A* decoding algorithm, which is widely used in large vocabulary continuous speech recognition (LVCSR) systems.

The rest of this paper is organized as follows. The descriptions of the statistical training of the unigram grapheme model and the n-gram grapheme model are given in section 2. A two-stage, fast decoding algorithm is presented in section 3. Some experimental results on our g2p system and the comparison with a baseline g2p system are presented in section 4. Finally, some discussion and conclusions are given in section 5.

2. Training of the grapheme model
A grapheme, or grapheme-phoneme joint multigram, is a pair comprised of a letter sequence and a phoneme sequence of possibly different lengths. For example, the word rough and its pronunciation / r ʌ ʊ f / can be represented by a set of three graphemes, i.e., [r], [u], and [f].

2.1 Training of the unigram grapheme model
Let us define:

\[ G \equiv \left\{ G_i \right\}_{i=1}^{\nu} : \text{set of training grapheme sequences} \]

\[ \Phi \equiv \left\{ \phi_j \right\}_{j=1}^{\nu} : \text{set of training phoneme sequences} \]
where $N$ denotes the size of the training pronunciation dictionary. A (map) grapheme model is a model in which the longest size of sequences in $G$ and $\Phi$ are $m$ and $n$, respectively. For example, a $(4, 4)$ grapheme model means that one phoneme with up to 4 letters and one phoneme can be grouped together to form graphemes. A joint segmentation (alignment) of $\vec{g}_i$ and $\vec{\phi}_i$ is given by:

$$\vec{q}_i \equiv (q_1, q_2, \ldots, q_L) = ([g_1, \phi_1], [g_2, \phi_2], \ldots, [g_L, \phi_L])$$

where:

$$\vec{g}_i = \{g_1, g_2, \ldots, g_L\} = \vec{g}_i$$

$$\vec{\phi}_i = \{\phi_1, \phi_2, \ldots, \phi_L\} = \vec{\phi}_i$$

$q_j \equiv [g_j, \phi_j], j = 1, 2, \ldots, L$ are the graphemes.

The unigram $(un)$ grapheme model parameter set $\Lambda$ is estimated using the maximum likelihood (ML) criterion:

$$\Lambda^* = \arg \max_{\lambda \in S(\vec{g}, \vec{\phi})} p(\vec{q}_i | \Lambda)$$

where $S(\vec{g}, \vec{\phi})$ is the set of all possible joint segmentations of $\vec{g}$ and $\vec{\phi}$. The parameter set $\Lambda$ is trained using the expectation-maximization (EM) algorithm. The EM algorithm is implemented using forward-backward algorithm to avoid the exhaustive search of all possible joint segmentation of grapheme sequences. A marginal trimming technique is used to eliminate the unigram graphemes whose likelihood are less than a certain threshold. During marginal trimming, the thresholds gradually increase from an initial small value to a larger value during each iteration of the training.

### 2.2 Joint segmentation of grapheme sequences

After we obtain the unigram grapheme model $\Lambda^*$, for each $(\vec{g}, \vec{\phi}) \in (G, \Phi)$, $i = 1, 2, \ldots, N$, we compute the optimal alignment using ML criterion:

$$q_i^* = \arg \max_{q_i \in S(\vec{g}, \vec{\phi})} p(q_i | \Lambda^*)$$

The optimal grapheme sequence $\vec{q}_i^*$ actually denotes the optimal joint segmentation (alignment) between the grapheme sequence $\vec{g}_i$ and the phoneme sequence $\vec{\phi}_i$, given the current trained unigram grapheme model.

### 2.3 Training of n-gram grapheme model

After we obtain the optimal joint-segmentation of grapheme and phoneme sequences, an n-gram language model of graphemes is constructed to model the contextual information between grapheme-phoneme sequences. For example, the grapheme $ough$ can be pronounced as $\text{ough}$, $\text{ough}$, and $\text{ough}$, as in words: rough, though, and thorough, respectively, depending on the context. In our system we use Cambridge-CMU statistical language model (SLM) toolkit to train the n-gram grapheme model. The priority of different backoff paths for tri-gram graphemes is listed in Table 1.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$P(C \mid A, B)$</td>
</tr>
<tr>
<td>4</td>
<td>$P(C \mid A, B) \ast BO_2(A, B)$</td>
</tr>
<tr>
<td>3</td>
<td>$P(C) \ast BO_1(B) \ast BO_2(A, B)$</td>
</tr>
<tr>
<td>2</td>
<td>$P(C \mid B)$</td>
</tr>
<tr>
<td>1</td>
<td>$P(C) \ast BO_1(B)$</td>
</tr>
</tbody>
</table>

In Table 1, $BO_1(B)$ and $BO_2(B)$ denote the backoff weights of trigram and bigram, respectively. In the decoding stage, the decoder always looks for the existing approximation of the n-grams with highest priority.

### 3 Decoding system

The basic idea behind the g2p decoder is to search the phone sequences that maximize the joint probability of grapheme sequences given the input word orthography sequence $\vec{g}$:

$$\vec{q}_i^* = \arg \max_{q_i \in S(\vec{g}, \vec{\phi})} p(q_i \mid \Lambda^*)$$

where $S(\vec{g}, \vec{\phi})$ denotes all possible phoneme sequences generated by $\vec{g}$, and $\Lambda^*$ denotes the n-gram grapheme model. The joint probability of a grapheme sequence given an n-gram grapheme model can approximately be computed as follows:

$$p(q_i) = p(q_{i_1} \cdots q_{i_L}) = \prod_{j=1}^{L} p(q_{i_j} \mid g_{i_{j-1}} \cdots q_{i_1})$$

In this paper, a fast, two-stage stack search algorithm is proposed to find the optimal pronunciation according to the criteria described in Eq. (5). The decoding algorithm consists of two steps which can be described as follows. 

#### Step-1

For an input word orthography sequence $\vec{g}$, search for the most likely grapheme sequences of the input word. The basic idea is to find the segmentation with furthest depth, while complying with the backoff priority defined in Table 1. Let us define depth $i$ as the current number of grapheme segments, and $\vec{g}_{i \mid \vec{g}_{i-1}}$ as the $i$-th grapheme sub-sequence at current depth $i$; $\vec{g}_i$ is the stack containing all possible grapheme sequences at current depth $i$. The algorithm can be summarized as follows:
while (not_end_of_word) do
    construct all possible valid n-gram graphone sequences
    \( g_{it+1, t+1} \) based on the elements of previous stacks
    and n-gram graphone model
    if \( p(g_{it+1, t+1}, S_{it+1}) \) exists then
        push \( g_{it+1, t+1} \) into \( g_s \)
    else
        search for backtrack paths with the priorities described in
        table 1; construct the new valid backtrack n-gram
        graphone sequences, and push them into \( g_s \).

    i++;

For example, consider the word *thoughtfulness*. The optimal
segmentation after step 1 of the decoding algorithm, for a (4,1)
graphone model with 3-gram SLM, is given by \( \{ \text{though}, t, f, u, l, n, e, s, s \} \).

**Step 2:** Given the optimal graphone sequences resulting from
step 1, search for the optimal phoneme sequences which will
maximize the joint probability of graphone sequences defined
in Eq. (6). Let us define \( n_{seg} \) as the number of graphone
segments resulting from step 1, \( n_i \) as the order of n-gram;
\( g_i \) as the i-th n-gram graphone in the graphone stack, \( \phi_i \) as all
possible n-gram phoneme sequences for graphone \( g_i \);
\( q_{il} \) denotes the phoneme constructed by graphone \( g_i \) and
phoneme sequence \( \phi_{il} ; p_{si} \) denotes the stack of current
phoneme candidates at depth \( i \).

for \( i \leftarrow 1 \) to \( n_{seg} \) do
    construct \( g_i = (g_{i1}, g_{i2}, ..., g_{in}) \)
    find all possible \( \phi_{il} \) from \( \Lambda_{g_i} \), construct \( q_{il} \)
    for \( k \leftarrow 1 \) to \( \phi_{il} \) do
        for \( l \leftarrow 1 \) to \( n \) do
            insert new phoneme token into \( p_{si} \)
            for each \( q_{il} \) allowed to follow \( \phi_{il} \), do
                update the graphone stack and the likelihood
                of each graphone sequence in the stack
            if \( p_{si} \) is unique then
                pop out \( ps_i \)
            else
                pop out the phoneme candidate with highest
                likelihood in the graphone stack;
                prune the stack.

Let us assume the average length of the word orthography and
the average number of phoneme mappings for each graphone
are \( M \) and \( N \), respectively. For each input word, the number of
possible graphone segmentations is exponential to the word
length. Furthermore, each graphone can map to multiple
phoneme entries in the pronunciation space, with different
likelihoods. As a result, the computational cost for direct
solution of the search problem defined in Eq. (5) is at
the order of \( O(M^N) \). On the other hand, step 1 of our
decoding algorithm only requires \( O(M) \) number of operations.

The step 2 of our decoding system requires \( O(N^{ve}) \) operations,
which is a nondeterministic polynomial (NP) problem. One
feature of our decoder algorithm is that it reduces a two-
dimensional exponential search problem into two one-
dimensional NP search problems, while still keep the
approximate optimization of Eq. (6).

4. Experimental results

4.1 Experiment setup

The experiments were conducted using SONY's internal
English pronunciation lexicon, which contains 66,000 entries.
Each entry contains one word orthography and its base-form
pronunciation, including primary and secondary stress
marks, and syllable boundaries. A random set of 56,000
entries was chosen for the training, and a disjoint set of 10,000
words was set aside for evaluation. We also compare the
performance of our g2p system with a previous decision tree
based g2p system described in [5]. The performance is
measured in three categories, namely phoneme accuracy,
memory consumption, and processing time.

4.2 Experimental results on the training of graphone model

In the training of the unigram graphone model, a marginal
trimming technique was applied to eliminate outliner or rarely
occurring graphones, and to significantly reduce the model
size. Table 2 shows the size of unigram graphone models
before and after the EM training, under different experimental
conditions, where \( (m,n) \) denotes the order of unigram
graphone model; \( e_{min} \) and \( e_{max} \) denotes the minimum
and maximum threshold for evidence trimming, respectively. We
can see from Table 2 that marginal trimming significantly
reduces the number of unigram graphones, which can be viewed as a fundamental unit to represent the mapping
between graphone and phoneme sequences.

The trained unigram graphone model is used to generate
alignments between words and their corresponding
pronunciations in the pronunciation dictionary. This alignment data
is used to generate n-gram graphone models in the second
stage of training. Table 3 shows the experimental results on
the size and perplexity of different n-gram graphone models.
The first three rows in Table 2 denote the experimental results
on trigram graphone model, and the last three rows denote the
results on bigram graphone models. The cutoff threshold is set
1 for all n-gram trainings.

<table>
<thead>
<tr>
<th>(( m,n ))</th>
<th>( e_{min} )</th>
<th>( e_{max} )</th>
<th>Size of graphone model set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (2,2) )</td>
<td>( 1e-6 )</td>
<td>1</td>
<td>106233 1641</td>
</tr>
<tr>
<td>( (2,2) )</td>
<td>( 1e-6 )</td>
<td>1e-3</td>
<td>106233 3326</td>
</tr>
<tr>
<td>( (4,1) )</td>
<td>( 1e-6 )</td>
<td>1e-2</td>
<td>78417 1931</td>
</tr>
<tr>
<td>( (4,1) )</td>
<td>( 1e-6 )</td>
<td>1e-3</td>
<td>78417 2028</td>
</tr>
<tr>
<td>( (3,3) )</td>
<td>( 1e-6 )</td>
<td>10</td>
<td>956513 908</td>
</tr>
<tr>
<td>( (3,3) )</td>
<td>( 1e-6 )</td>
<td>1</td>
<td>956513 5206</td>
</tr>
</tbody>
</table>
Table 3. Experimental results on n-gram grapheme model.

<table>
<thead>
<tr>
<th>(m,n)</th>
<th>$\alpha_{\text{uni}}$</th>
<th>$\alpha_{\text{tri}}$</th>
<th>Size of grapheme model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,1)</td>
<td>e-6</td>
<td>e-2</td>
<td>1777 8342 29854</td>
<td>15.94</td>
</tr>
<tr>
<td>(2,2)</td>
<td>e-6</td>
<td>e-1</td>
<td>3189 21818 37702</td>
<td>42.64</td>
</tr>
<tr>
<td>(2,2)</td>
<td>e-6</td>
<td>e-0</td>
<td>3097 22531 N/A</td>
<td>43.11</td>
</tr>
<tr>
<td>(4,1)</td>
<td>e-6</td>
<td>e-2</td>
<td>1777 8342 N/A</td>
<td>27.34</td>
</tr>
<tr>
<td>(2,2)</td>
<td>e-6</td>
<td>e-1</td>
<td>3189 21818 N/A</td>
<td>79.51</td>
</tr>
<tr>
<td>(2,2)</td>
<td>e-6</td>
<td>e-0</td>
<td>3097 22531 N/A</td>
<td>81.22</td>
</tr>
</tbody>
</table>

We can see from Table 3 that we can use a rather small set of unigram ($O(1^2)$) and 2-gram or 3-gram ($O(1^3)$) graphemes for English. Since the size of the n-gram models are rather small, this also enables a fast and efficient decoding algorithm for g2p conversion for different applications of speech processing. Another observation is that the perplexity of the grapheme model almost doubled as we reduced from the trigram model to the bigram model.

4.3 Evaluation results on phoneme accuracy and memory consumption

In the evaluation stage, we measure the recognition accuracy, memory requirement, and the processing time of our new grapheme based g2p system and compare it with a decision tree based g2p system (baseline) described in [5]. Our g2p system is able to automatically generate a pronunciation dictionary with different options, such as both primary and secondary stress markers (for TTS) or without any stress marker (for dynamic speech recognition). Our system can also generate baseform pronunciations in SAMPA-IPA format or CMU phoneme format. In the training and decoding of the g2p system, vowels with different stress markers are treated as different phonemes, i.e., low1, low2, and lo2. This results in 71 different English phonemes with 26 consonants and 45 vowels with different stress markers. In our experiments, the unigram and n-gram grapheme models are trained with labels of both primary and secondary stress markers. All of the evaluation experiments were conducted on the same FC with a Pentium 4, 2GHz CPU, and Linux Redhat 9.0 OS. The memory consumption and the processing time are measured for the entire 10k testing vocabulary.

Table 4 compares the performance of our grapheme-based g2p system under different experiment conditions versus the baseline g2p system described in [5]. Here #str denotes the number of stress markers, acc means the phoneme accuracy, corr means the word correct, mem denotes the memory consumption for the decoding of entire 10k testing words, and Spd means the processing time in seconds. We can see from Table 4 that our g2p system has similar recognition accuracy as the baseline. On the other hand, it only consumes 1.9 to 1/15 memory and is around 10 times faster when compared to the decision tree based system. We can also see from Table 4 that additional stress markers will increase the phone error rate by 3% - 4%, although further automatic labeling of secondary stress markers will only marginally increase the phoneme accuracy by less than 1%.

Table 4. Performance of grapheme based g2p system.

<table>
<thead>
<tr>
<th>(m,n)</th>
<th>$\alpha_{\text{uni}}$</th>
<th>$\alpha_{\text{tri}}$</th>
<th>#str</th>
<th>acc (%)</th>
<th>corr (%)</th>
<th>mem (MB)</th>
<th>Spd (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,1)</td>
<td>e-6</td>
<td>e-2</td>
<td>0</td>
<td>91.3</td>
<td>53.4</td>
<td>3.56</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>87.8</td>
<td>44.7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>87.5</td>
<td>44.1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(2,2)</td>
<td>e-6</td>
<td>e-1</td>
<td>0</td>
<td>90.5</td>
<td>49.8</td>
<td>6.34</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>87.2</td>
<td>41.7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>86.9</td>
<td>41.1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td></td>
<td></td>
<td></td>
<td>88.4</td>
<td>50.1</td>
<td>55.6</td>
<td>134</td>
</tr>
</tbody>
</table>

5. Conclusion

In this paper, an efficient, data-driven approach was proposed for g2p conversion. Our g2p system is capable of obtaining similar phoneme recognition accuracy while requiring around 1/10 of system memory and 1/10 of processing time when compared against a decision tree based g2p system. There are several reasons why our g2p system is more efficient than the inductive learning based approach. First, instead of automatically learning many redundant rules, we obtained a compact representation of the internal mapping between the grapheme and phoneme sequences using grapheme model. The size of the grapheme model was significantly reduced through marginal trimming during the training stage. Second, instead of directly using the existing search algorithm in LVCSR, such as the A* algorithm, an efficient, two stage decoding algorithm was proposed in our work.

6. Acknowledgements

We want to thank Erika Kobayashi, Tomoki Nitta, and Makoto Akahane from the speech lab of SONY NCE, Tokyo for their sponsorship, and assistance on this project.

7. References

[7] URL of Naval Research Lab text to phoneme rules, file://svr.gplv.cern.ac.uk/pub/compspeech/syntheses/english2phoneme.tar.gz