DATA-DRIVEN LEXICAL MODELING OF PRONUNCIATION VARIATIONS FOR ASR

Qian Yang and Jean-Pierre Martens
ELIS-RUG, Sint-Pietersnieuwstraat 41, B-9000 Gent (Belgium)
{qianyang,martens}@elis.rug.ac.be

ABSTRACT
In this paper a method for the automatic construction of a lexicon with multiple entries per word is described. The basic idea is to transform a reference word transcription by means of stochastic pronunciation rules that can be learned automatically. This approach already proved its potential (Cremelie & Martens, 1999), and is now brought to a much higher level of performance. Relative reductions of the word error rate (WER) of 20 % (open vocabulary) to 45 % (closed vocabulary) are now within reach.

1. INTRODUCTION
It is generally acknowledged that ASR can still benefit from improvements at all processing levels: feature extraction, acoustic, lexical and language modeling. Most of the benefits so far came from the acoustic level, e.g. by introducing dynamic features, context-dependent phone models and state tying on the basis of decision trees. The advances in language modeling and lexical modeling were much less rewarding. Nevertheless, most modern recognizers use a partly handcrafted lexicon comprising multiple pronunciations of frequent words (Lamel & Adda, 1996), and there is a growing interest in automatic pronunciation modeling for ASR (see Strik & Cucchiarini (1999) for an overview).

In this paper we propose a methodology for the automatic development of a lexicon comprising multiple pronunciations per word. Section 2 describes the general outline of this approach, sections 3 to 5 the details, and section 6 the evaluation experiments that were carried out.

2. GENERAL OUTLINE OF THE APPROACH
The basic ideas underlying our approach are (1) that word pronunciation variants can be obtained by transforming the reference transcription with multiple entries per word is described. The basic idea is to transform a reference word transcription by means of stochastic pronunciation rules, and (3) that these rules can be learned automatically from an orthographically transcribed speech corpus.

2.1. Stochastic pronunciation rules
The stochastic pronunciation rule format is borrowed from Cremelie & Martens (1997). If at some place in the reference transcription a focus pattern $F$ is found, surrounded by left and right context patterns $L$ and $R$, then the rule

$$\tau : LF \rightarrow F'$$

is said to match the reference at that place. It can produce two pronunciations there: one with $F$ transformed to $F'$ (probability $P_{F'}$) and one with $F$ left unaltered (probability $1 - P_{F'}$). The variable length patterns $L$, $R$ and $F$ together constitute the rule condition, and $(F, F')$ the transformation associated with the rule.

Rules with a low firing probability describe situations in which a transformation is mostly forbidden. They can be interpreted as

$$\tau : LF \rightarrow \neg F' \quad \text{with} \quad 1 - P_{F'}$$

and are therefore called negative rules. They often describe exceptions to more general positive rules with a high firing probability. It is an important property of our rule learning methodology that it can learn both positive and negative rules at the same time.

2.2. Rule hierarchy
The pronunciation rules cannot be considered independently of each other. Together they constitute a complex operator which is governed by a rule hierarchy for resolving the following issues: (1) what rule to select when different rules can be used to perform the same transformation, and (2) what transformation to select when different transformations can be performed?

In order to address the first question, the rules associated with a particular transformation are ranked according to the length of their condition, with the longest one on top of the list. Our strategy is to select at each examined position only the first matching rule for performing this transformation. This ensures that a negative rule can prevent a more general rule from performing the same transformation with a higher probability.

In order to address the second question, we select one rule for each transformation that is possible at a certain position, and we rank these rules according to the following criteria (in order of preference): the condition length, the focus length, the difference in length between $F$ and $F'$, $F'$. Then we determine for each rule the chance that none of its predecessors has fired, and we let the rule fire with its firing probability multiplied by that chance.

3. LEARNING THE RULES
Learning rules from a corpus of orthographically transcribed utterances is accomplished in four phases.

3.1. Phase 1: Generating training examples
By concatenating the right reference word transcriptions, one obtains a reference transcription of the utterance. By aligning the speech with a stochastic automaton modeling the reference transcription and the phoneme acoustics, one creates an expert transcription which is supposed to represent the correct pronunciation of that utterance (Cremelie & Martens, 1997).

Both transcriptions are ligned-up with one-another and word boundaries (symbol '%') are inserted in order to allow for a distinction between word internal, word boundary and cross-word
In practice, the investigation of rule $C$ goes like this: 

For each transformation with an admissible focus, the first matching rule is selected. Then, all these selected rules are ranked according to the rule hierarchy.

In order to determine the firing probabilities, two counters $n_1(\tau)$ and $n_2(\tau)$ are associated with each rule. For updating these counters, the reference transcription is scanned from left to right and per position $p$, the following operations are performed:

1. Identify the transformation $(F, F')$ (if any) to perform.
2. If an invalid transformation is discovered, ignore it (increment $p$ by the focus length), in any other case:
   a) Construct the list of selected rules, set $executable$ = true.
   b) As long as $executable$ = true, do the following for each selected rule: increment its $n_1$ and if its transformation is the one to perform, then increment its $n_2$, set $executable$ = false and add length $(F)$ to $p$.
   c) Increment $p$.

The firing probability is then given by

$$ P_f(\tau) = \frac{n_2(\tau)}{n_1(\tau)}.$$
1. Identify the parent $P$ with the smallest $\Delta H_{cp}$.
2. If rule $C$ is selected less than $N_{rs}$ times, or if $\Delta H_{cp}$ is smaller than $D_{cp}$, then remove $C$ and add its counters to those of $P$.
3. If $C$ was removed and the identified parent $P$ was just the second parent of $C$ in the rule list, then interchange it with the first parent of $C$.

The process is controlled by two variables: $N_{rs}$ and $D_{cp}.$

## 4. THE VARIANT GENERATION PROCESS

The variant generation process applies the rules to create pronunciation variants of a single word. These variants will all together constitute the pronunciation model of that word. It is important to make the variant generation compatible with the rule learning process. This is not trivial given that the latter one operates on whole utterances rather than on single words.

### 4.1. Context-specific variant lists

When there are cross-word rules (defined as having a condition extending over a word boundary), the variant generation process will first produce a variant list for every relevant word context, and then compile all lists in one pronunciation model. Suppose that the word is *has* (reference transcription /h a z/) and that the cross-word rules are

- **r1:** $n \% h a z \rightarrow \sim$
- **r2:** $@ r \% r \rightarrow \sim$
- **r3:** $\% s \rightarrow \sim$

Together with the empty context, the left context of $r_1$ and the right context of $r_1$ must be considered as potential left/right word contexts. This leads to 4 so called context-dependent extended references for the given word:

- **r1:** $n \% h a z \% @ \% * * * *$
- **r2:** $@ r \% r \% * * * *$
- **r3:** $\% s \% * * *$

with the stars being added to obtain word contexts of the maximum length of a cross-word rule context (which is $N_{lr} + 1$). During the variant list generation, the stars are assumed to match with nothing else. This way, all variants will be created assuming that just the specified non-star context part is present.

Note that for building pronunciation models for a recognizer with a given lexicon and language model, one only needs to generate variant lists for word contexts that can occur given these extra restrictions.

### 4.2. Variant list generation

Per extended reference, a list of variants will be created. It is initialized with one variant, the extended reference. Gradually, more and more variants are added by letting selected rules act on existing variants at different positions. During this process, each variant has a head resulting from former operations, and a tail that is susceptible to further transformations. The tail is a copy of the reference tail, and its initial phoneme is marked by a variant pointer $p_v$, its corresponding position in the reference by a reference pointer $p_r$.

Once the pointers and the probability of the initial variant are filled in, the extended reference is scanned from left to right, and for every position $p$ the following operations are performed:

1. Check whether there are variants whose $p_v = p$. If there are, construct the list of selected rules $\{ \{ r_{pi} : F_{pi}, F'_{pi} \} \}$.
2. For each existing variant whose pointer $p_v$ is equal to $p$:
   - Set $P_{novar} = P_v$, the probability of this variant.
   - For every selected rule $r_{pi}$:
     - Compute $P_{novar}$ as $P_{novar} \cdot P_{f_{io}}[r_{pi}]$
     - If $P_{novar} \geq P_{min}$, let the rule fire to create a new variant (perform associated transformation, attach a probability $P_{novar}/P_v$ to phoneme $p_v$, add length($F_{pi}$)+1 to $p_v$, add length($F'_{pi}$)+1 to $p_v$, set new variant probability to $P_{novar}$), and multiply $P_{novar}$ by $1 - P_{f_{io}}[r_{pi}]$.
   - If $P_{novar} < P_{min}$, then remove the existing variant from the variant list.

4.3. Compiling the word pronunciation model

A variant list can be represented by a tree shaped network like the one shown on Figure 2. The left and right word contexts are

$P_v = P_s \prod_{i=1}^{t-1} \left( \prod_{k=1}^{i} (1 - P_{f_{io}}[r_{pi}]) \right)$

Clearly, $P_{min}$ controls the number of variants being produced.

### 4.4. INTEGRATING THE MODELS IN A RECOGNIZER

A word pronunciation model can formally be described as a black box characterized by a reference transcription and a number of inputs/outputs that are labeled with the condition to be met by the connecting word. If the recognizer wants to move from word $W_i$ to word $W_j$, it can only do so via the output of $W_i$ whose condition best matches the reference transcription of $W_j$ (count number of matching phonemes), and via the input of $W_j$ whose condition best matches the reference transcription of $W_i$.

6. EXPERIMENTS ON TIMIT

All experiments described in this section were carried out on the TIMIT database (Lamel et al., 1986). It contains spoken utterances of 462 training and 168 test speakers: 3150 SX-sentences (5 per speaker) taken from a limited set of 450 different phonetically rich sentences with a vocabulary of 1793 words, and 1890 SI-sentences...
(3 per speaker) being all different and with a vocabulary of 5143 words. The total vocabulary (SX+SI) is 6224 words. A bigram language model was derived to model all the TIMIT sentences (Cremelie & Martens, 1999).

6.1. System configuration
The experiments were carried out with a segment-based recognizer incorporating context-independent acoustic models that were trained discriminatively (Verhasselt & Martens, 1998). The lexicon was either composed of the reference word pronunciations provided with the database, or the pronunciation models created by our method. In both cases, the pronunciation models were supplemented with error arcs (allowing for deletion, insertion and substitution of phonemes) (Cremelie & Martens, 1997) characterized by error probabilities that can be trained automatically. In fact, in combination with the reference transcription, these arcs represent context-independent pronunciation rules of the form \( F \rightarrow F' \) with \( F \) and \( F' \) being either single phonemes or empty.

The pronunciation rules were trained on the SX training sentences and tested on all sentences of the core test set. The control variables \( N_F \) and \( N_{\text{LR}} \) were fixed to 5 and 2 respectively.

6.2. Baseline system
The baseline system (BS) is the same one that was mentioned in Cremelie & Martens (1999). It incorporates the lexicon supplied with the database. Its performances on the different parts of the core test set are listed in Table 1. That the WERs are not exactly the same as those mentioned in Cremelie & Martens (1999) is due to the fact that the procedure for training the error probabilities was slightly modified. Although this did not cause any drop in the SX+SI error rate, it did cause an improvement of the SX results (which is the closed vocabulary case).

6.3. Rule learning experiments
It was found that 99.4 % of the pronunciation variations observed in the training data can be explained by 659 different valid transformations. By setting \( N_{\text{trans}} = 5 \), still 90 % of these variations can be explained by just 156 transformations.

By changing \( N_{\text{rs}} \) and \( D_{\text{rp}} \), one can control the size of the rule set that is retained. For \( N_{\text{trans}} = 5 \), it was found that increasing \( D_{\text{rp}} \) resulted in a smooth reduction of the rule set size, provided \( N_{\text{rs}} \) is sufficiently large (\( \geq 8 \)).

6.4. Variant generation experiments
The average number of variants per word is almost completely determined by \( P_{\text{min}} \). It does not seem to depend much on the number of retained pronunciation rules.

6.5. Recognition experiments
The performance of the recognizer before and after the automatic retraining of the error probabilities was measured for different combinations of \( D_{\text{rp}} \) and \( P_{\text{min}} \) (with \( N_{\text{trans}} = 5 \), \( N_{\text{rs}} = 10 \)).

For \( P_{\text{min}} = 0.05 \), the recognition performance was measured as a function of the rule set size (see Table 2). The average number of variants per word ranged from 2.6 to 2.9. The improvements over the baseline system are substantial and pretty stable as a function of \( D_{\text{rp}} \).

<table>
<thead>
<tr>
<th>( D_{\text{rp}} )</th>
<th>nr of rules</th>
<th>SX+SI</th>
<th>SX</th>
<th>SI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2289</td>
<td>6.46</td>
<td>3.08</td>
<td>11.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.28</td>
<td>2.83</td>
<td>10.89</td>
</tr>
<tr>
<td>0.005</td>
<td>1181</td>
<td>6.52</td>
<td>3.29</td>
<td>10.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.15</td>
<td>2.12</td>
<td>11.59</td>
</tr>
<tr>
<td>0.020</td>
<td>628</td>
<td>6.76</td>
<td>2.76</td>
<td>12.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.22</td>
<td>2.55</td>
<td>11.16</td>
</tr>
</tbody>
</table>

Table 2. Recognition performances (WER in %) as a function of the rule set size. Per size both the results before (top) and after (bottom) error probability retraining are provided.

For two cases (1181 and 2289 rules), the recognition performance was also measured as a function of the average number of variants per word (Figure 3). For the closed vocabulary case (train and test on SX), we find relative improvements of 30...45 % with respect to the baseline. This is significantly more than the 15 % reported in Cremelie & Martens (1999).

6.6. Conclusion
The results clearly show that by automatic retraining of the error probabilities, it is possible to improve the recognition performance of a hybrid system considerably. This is especially true for closed vocabulary situations. The improvements are obtained by only adding a small number of pronunciation rules. This shows that the automatic retraining of the error probabilities is an effective way to improve the performance of a pronunciation system.

7. REFERENCES