AN APPROACH TO LOW FOOTPRINT PRONUNCIATION MODELS FOR EMBEDDED SPEAKER INDEPENDENT NAME RECOGNITION

Kaisheng Yao and Lorin Netsch

Speech Technologies Laboratory, Texas Instruments Incorporated
12500 TI Blvd, MS 8649, Dallas, TX 75243
{kyao, netsch}@ti.com

ABSTRACT

Pronunciation modeling is an important component of speaker independent name recognition on embedded devices. Decision trees have been widely used to generate pronunciations of names due to improved accuracy. However, pronunciation modeling using decision trees may suffer from two main draw backs. The first is large memory footprint. The second is that decision trees usually generate a single pronunciation which does not reflect the real-world multiple pronunciations of a name. We present an approach to address these draw backs. The approach consists of a letter-to-phoneme mapping method that prunes many irregular pronunciations in order to train compact decision trees, and a multi-stage pronunciation transformation method that generates multiple pronunciations from the output of the trained decision trees. The approach effectively reduces footprint by more than 58% and achieves more than 23% of word error rate reduction, compared to a baseline.

Index Terms— Speech recognition, pronunciation model, decision tree, probabilistic re-write rule

1. INTRODUCTION

An important application of automatic speech recognition (ASR) on embedded devices is speaker independent name recognition. It is characteristic for the application that the user does not have to train a name tag for each name. Furthermore, vocabularies of the names are subject to change by the user. Therefore, the phoneme sequences of the names need to be generated on-line. A pronunciation model that performs this function is thus a critical component of the name recognition system.

In developing a pronunciation model, several approaches have been proposed. One approach applies a set of rules to map the orthographic form of words to pronunciations. The rules usually have very low footprint and hence are used in some early name recognition systems [1]. However, the rules may not be sufficient to model irregular languages, such as English, accurately enough. More recently, decision trees (DTs) [2] have been applied for modeling pronunciations of irregular languages [3]. The DT-based approach is usually more accurate and provides better recognition accuracy than the rules. However, simple application of DTs may have large memory footprint, which is not feasible for embedded ASR. Another drawback of the DTs is that they usually produce a single pronunciation of each word, which inevitably introduces mismatch to real-world multiple pronunciations of names.

This paper describes an approach developed in our lab to improve DT-based pronunciation models. Since the available memory is limited on embedded devices, a method is developed to regularize a pronunciation dictionary to achieve a low footprint. In addition, performance of the DTs may be improved by using a set of data-driven rules [4] to augment pronunciations from the DTs. Notice that these rules usually require little memory. Hence performance can be improved without much increase of memory footprint. Experimental results show that our approach can significantly reduce memory footprint and improve recognition performance, in comparison to a baseline system.

2. THE PRONUNCIATION MODELING APPROACH

2.1. Pronunciation modeling using decision trees

Our embedded name recognition system consists of three major units: pronunciation models, acoustic models, and a recognition engine. For the name recognition task, names are initially presented in written form. Phonetic transcriptions are generated by the pronunciation model. The recognition engine carries out the recognition process with the phonetic transcription and the acoustic model.

It is critical to have accurate phonetic transcriptions in order to achieve high performance name recognition. Decision trees have shown the highest accuracy for letter-to-phoneme (LTP) mapping [5], and hence are widely used [3]. Using decision trees, pronunciation of each letter in the alphabet of a language is modeled separately. Pronunciation of a name is obtained by concatenation of pronunciations of letters in the name.

However, for irregular languages such as English, simple application of DTs may result in large memory footprint to encode the LTP mapping. An example is shown below. For instance, a reasonable one-to-one LTP mapping between word `TROPHIES` and its pronunciation `t r ow f iy z` may be

```
T R O  P H  I  E  S
t r ow f _  iy _  z
```

This word is not unusual for English in having fewer phonemes than letters. The null-phone _ needs to be inserted in the transcription if a one-to-one mapping is to be maintained. Yet it is not clear where the null-phone should be placed, since the following may also be a reasonable mapping:

```
T R O  P H  I  E  S
t r ow f _  _ iy  z
```

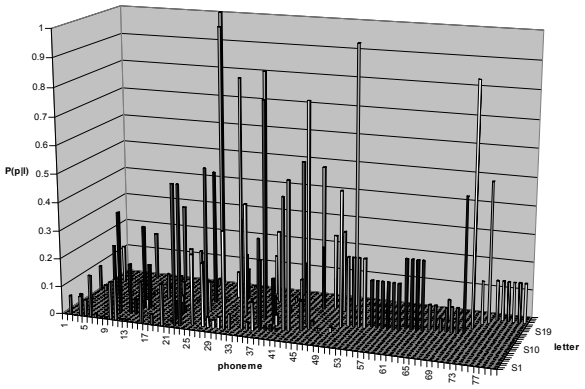

Fig. 1. Estimated posterior probability of phoneme given letter.

2.2. The proposed LTP method

Based on the above observation, a method of regularizing LTP mappings is proposed. The concept of regularization in this paper is to remove those LTP mappings with low probabilities using a pruning process. The LTP mappings after the regularization are those with sufficient probabilities and are used to train decision trees, generating compact and low footprint pronunciation models.

Define p and l as phoneme and letter, respectively, in a pronunciation dictionary. The above mentioned probability is the *a posteriori* probability of phoneme p given letter l, i.e., $P(p|l)$. Naturally, this is obtained via the Bayes formula $P(p|l) = \frac{P(l|p)P^{\star}(p|l)}{P(l)}$. In the Bayes formula, the probability of observing letter l given phoneme p, $P(l|p)$, is measured as the frequency of occurring letter l given phoneme p. Clearly, alignments of letter-to-phoneme pairs have to be obtained to count the co-occurrence count of phoneme p and letter l and the count of phoneme p, from which the frequency can be calculated. The prior probability $P^{\star}(p|l)$ is usually initialized with human knowledge. The probability of letter l, $P(l)$, is a normalization factor in the Bayes formula.

For example, setting an initial small value of the prior probability of phoneme `iy` given letter `E`, $P^{\star}(\texttt{iy}|\texttt{E})$, may result in low posterior probability $P(\texttt{iy}|\texttt{E})$. However, if a large value of $P(\texttt{E}|\texttt{iy})$ is observed on the aligned letter-to-phoneme pairs, the posterior probability $P(\texttt{iy}|\texttt{E})$ may become higher.

This example shows the necessity of an iterative process to update the *a posteriori* probabilities. In this paper, an E-M type algorithm is applied. In the E-step, the *a posteriori* probability is obtained via the above mentioned Bayes formula. In the M-step, the updated posterior probability is used together with the letter probability $P(l)$ to obtain the best LTP mapping.

The above E-M algorithm is iteratively conducted. At convergence, some *a posteriori* probabilities become zero. Strong peaks occur for strong co-occurrences of certain phonemes and letters. An example of the converged posterior probability $P(p|l)$ is shown in Fig. 1, which has strong co-occurrence of phoneme `ax` and letter `A` [1].

We use entropy to measure irregularity of LTP mapping. Averaging over all letter-to-phoneme pairs in an English pronunciation dictionary, we obtained the averaged entropy at initialization as 0.78. After 5 iterations, the averaged entropy decreased to 0.57. This quantitative result showed that the LTP method was able to regularize LTP mappings.

[1] We use numeric indices in Fig. 1.

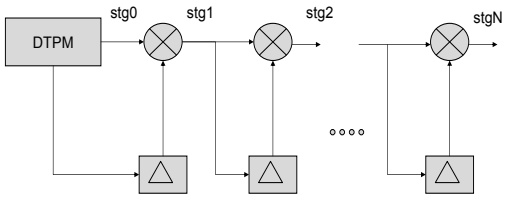

Fig. 2. The diagram of the multi-stage pronunciation adaptation method of generating multiple pronunciations.

Whereas some related methods [6] may terminate after the above process, the proposed method further applies a pruning process on the posterior probabilities. A threshold θ_{LTP} is selected to prune those phoneme and letter co-occurrences that have low posterior probabilities. For example, posterior probability of phoneme `w_ah` given letter `A` is lower than the threshold and therefore the co-occurrence of `A` and `w_ah` is removed. After the pruning process, another EM iteration is run to align letter-to-phoneme pairs, providing training cases for decision trees [2].

2.3. A multi-stage method to generate multiple pronunciations

Decision trees trained with the above aligned pronunciation dictionary usually output a single pronunciation for a name. However, high performance name recognition requires a system to deal with multiple pronunciations. Hence, a single pronunciation of decision trees has to be augmented with its pronunciation variations.

In this section, we present a method to generate multiple pronunciations through stages of transformations. A diagram of the proposed method is shown in Fig. 2. Single pronunciations from the trained decision trees, denoted as `DTPM` in Fig. 2, are transformed at the first stage. The transformation, denoted as $\bigotimes$, applies a set of probabilistic re-write rules [4] at this stage. After the transformation, the outputs of this stage consist of the original input pronunciations and their variations due to the rules. The outputs are then used as inputs to the next stage, and the above process is applied again.

The rules in each stage, denoted as $\triangle$ in the figure, are obtained offline. In order to obtain the rules, the patterns of pronunciation variation have to be determined after analyzing the alignment between two sets of pronunciation dictionaries. The first dictionary, reference dictionary, consists of *true* pronunciations, and the second consists of pronunciations generated from the previous stage. With the selected input pronunciation, a pattern of pronunciation variation from the reference pronunciation is extracted. The pattern includes the preceding and following phoneme context of the pronunciation variation. The word boundary, denoted as `$` is also considered as a context. The phoneme to be transited is denoted as `_` in a context.

For each pronunciation variation, the patterns are organized into a tree structure. Each node represents a context. The context-dependent phoneme transition probability $P(p \rightarrow p'|c)$ is calculated similarly as that in Sec. 2.2 using the co-occurrence counts of phoneme p and its variant p' given a context c. An example of the tree is shown in Fig. 3. In the figure, phoneme `ah` may have a pro-

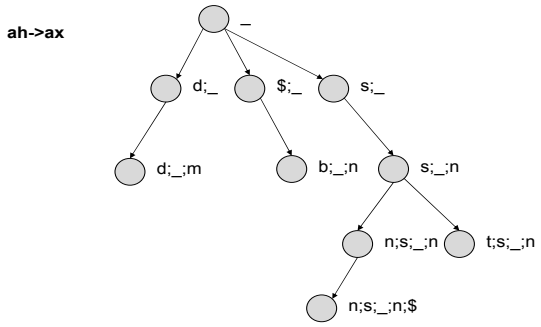

Fig. 3. Tree-structured rewrite rules for phoneme variation pattern `ah` to `ax`. `_` denotes the phoneme to be transited, which is `ah` in this figure. The top node is context-free. The deepest node has the longest context.

nunciation variation `ax` if, e.g., the left phoneme of phoneme `ah` is `d`. More detailed analysis shows that the pronunciation variation occurs when the right phoneme is also `m`.

These contexts are grouped into several rule sets with different lengths of left and right contexts. Denote a rule set with left and right context lengths as R_{ij}. For example, the contexts `n;s;_;n` and `t;s;_;n` in Fig. 3 are grouped into rule set R_{21}.

Reliable rule sets are kept using a pruning process [7]. After pruning, the context selected to transit a phoneme to its variant may not be very detailed as those in the bottom of the tree, e.g., R_{22}, nor too general as those in the root of the tree, i.e., R_{00} which is context free. For example, the contexts selected for transition of `ah` to `ax` shown in Fig. 3 are in rule set R_{10}, which includes contexts such as `d;_`, `$;_`, and `s;_`.

Pronunciation variations in each stage are generated by applying the pruned rule set. When a context c is located for a phoneme p, its variant p' is generated with probability for word W, i.e.,

$$P(p'|W) \leftarrow P(p|W)P(p \rightarrow p'|c).$$

This probability $P(p'|W)$ has to be larger than a threshold to transit phoneme p to p' for word W. We also keep the original pronunciations in the output of each stage.

3. EXPERIMENTAL RESULTS

3.1. Experiment setup

The Wall Street Journal (WSJ) database was used to train acoustic models. The CALLHOME American English Lexicon (PRONLEX) [8] is used to train the DT-based pronunciation model. Since our task is name recognition, which may not have letters such as `.` and `'`, we removed those entries from the dictionary. We also augmented the dictionary with additional words. The final dictionary has 96500 entries with multiple pronunciations.

Our name recognition system was tested on an in-house proprietary database collected using hands-free microphones in three recording sessions: parked (car parked, engine off), city-driving (car driven on a stop and go basis), and highway (car driven on a highway). In each session, 20 speakers (10 male/female) read 120 pairs of English first and family names. The database was sampled at 8kHz, with frame rate of 20ms. From the speech, 10-dimensional MFCC features, together with their first-order differentials, were derived. A joint additive and convolutive distortion compensation (JAC) method [9] was used to enhance performance in the above three driving conditions.

Table 1. Relative reduction of memory footprint (in %) and phoneme accuracy (PA in %) of the pronunciation model using different pruning threshold θ_{LTP} and prior probabilities.

$P^\star(p\|l)$	θ_{LTP}	0.000	0.001	0.003	0.005
	% Reduction	58	59	60	62
	PA	83.7	88.5	88.6	88.4
Better	% Reduction	60	61	61	61
a priori	PA	88.7	88.6	88.7	88.7

Performance of the pronunciation model was evaluated on the database. Phoneme accuracy (PA), which is shown below, is calculated by comparison of the pronunciations generated from the pronunciation model and the pronunciations in a reference dictionary of the database. The PA is defined as

$$Phoneme\ accuracy = \frac{N - D - S - I}{N},$$

where N is the total number of phonemes in the reference pronunciations. D, S and I each denote the number of deletion errors, substitution errors and insertion errors. The contribution of the pronunciation model to speech recognition is measured in relative word error rate (WER) reduction.

3.2. Memory reduction

The memory footprint is computed on the trained DTs. An initial implementation of DTs uses an LTP mapping that does not have the method in Sec. 2.2. DTs are then trained with the pronunciation dictionary processed by the LTP method in Sec. 2.2, which incorporates prior knowledge and uses a set of pruning thresholds θ_{LTP} to regularize the pronunciation dictionary. As shown in Table 1, memory footprint is reduced by 58%, in comparison to the initial DTs.

Further performance improvements may be obtained by incorporating better prior knowledge. In particular, we adjusted $P^\star(p|l)$ of those unlikely pairs to zero. We observed that footprint is reduced by 60% with $\theta_{LTP} = 0.000$. However, the contribution of better prior knowledge is diminishing when the pruning threshold θ_{LTP} is applied. With θ_{LTP} increased from zero to 0.005, pronunciation models with different prior probabilities tend to have similar footprint and phoneme accuracies. The results show that pruning in the LTP method is effective to reduce irregularity in the aligned dictionary.

3.3. Improved pronunciation using the multi-stage method

3.3.1. Results on phoneme accuracy

The multi-stage method reported in Sec. 2.3 is applied on the output of the DTs. Phoneme accuracy as a function of the stage number is shown in Fig. 4. The phoneme accuracy is increased after each processing stage. Notice that the first stage of the proposed

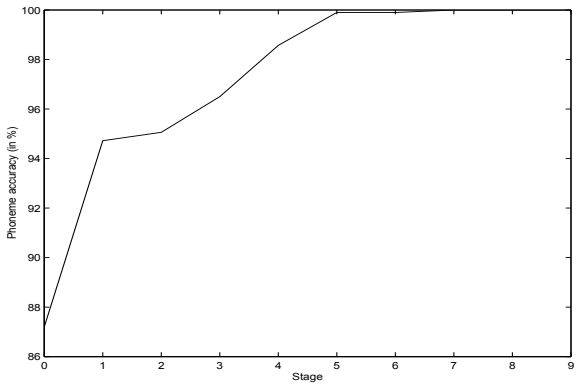

Fig. 4. Phoneme accuracy versus stage index.

Table 2. Number of data-driven rules at each stage of the proposed method.

Stage	1	2	3	4	5	6	7	8	9
#	183	135	107	97	92	87	86	85	83

method is able to increase phoneme accuracy by 8%. Improvements of phoneme accuracies are from 0% to 2% in the following stages. After the 6-th stage, phoneme accuracy is at 100%, meaning that all of the multiple pronunciations of names are included in the generated pronunciations. We also show the number of the extracted probabilistic re-write rules at each stage in Table 2. The number of rules is decreased from 183 to 83.

These results clearly show that each stage extracts rules that reduce pronunciation variation between the input pronunciations and their reference pronunciations. Since the number of the extracted rules at each stage is small, extra memory to save these rules is small.

3.3.2. Results on name recognition

Relative word error rate reduction is measured against the stage 0, which has single pronunciations of names. Table 3 shows the reduction as a function of the stage number. We observe that

- With multiple pronunciations generated in the stage 1, WER is significantly reduced by 23%, relative to the stage 0.
- Additional multiple pronunciations in the following stages contribute to further WER reductions. In the stage 4, WER is reduced by 32%. However, we notice that the major reduction of WERs is achieved in the stage 1.
- We conducted some analysis in each driving condition and observed more significant reduction of WERs in the parked condition than in the highway condition. Notice that the major mismatch in the parked condition may be caused by pronunciation variation, whereas distortions in the highway condition include much background noise. In the parked condition, a significant 61% of WER reduction is observed. The results show that the method reduces much mismatch caused by pronunciation variations between DTs and real pronunciations of users.

Table 3. Relative word error rate reduction (in %) of name recognition achieved by the proposed method.

stage	0	1	2	3	4
WER reduction	0	23	30	30	32

4. CONCLUSIONS

In this paper, we have presented an approach to improve decision-tree-based pronunciation model for speaker-independent name recognition. The approach regularizes a pronunciation dictionary to train compact decision trees. To reduce mismatch of the single pronunciation output from the decision trees, a multi-stage method is applied to augment the pronunciation with its variations. On a hands-free name recognition task performed on an embedded ASR system, we obtained significant performance improvements using this approach, compared to a baseline system.

5. REFERENCES

[1] C. Ramalingam, L. Netsch, and Y. Kao, "Speaker-independent name dialing with out-of-vocabulary rejection," in *ICASSP*, 1997, pp. 1475–1478.

[2] J. Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers, 1992.

[3] J. Suontausta and J. Tian, "Low memory decision tree method for text-to-phoneme mapping," in *ASRU*, 2003.

[4] Q. Yang and J.-P. Martens, "Data-driven lexical modeling of pronunciation variations for ASR," in *ICSLP*, 2000.

[5] V. Pagel, K. Lenzo, and A. Black, "Letter to sound rules for accented lexicon compression," in *ICSLP*, 1998, pp. 2015–2018.

[6] R. Damper, Y. Marchand, J.-D. Marsters, and A. Bazin, "Aligning letters and phonemes for speech synthesis," in *ISCA Speech Synthesis Workshop*, 2004.

[7] Y. Akita and T. Kawahara, "Generalized statistical modeling of pronunciation variations using variable-length phone context," in *ICASSP*, 2005.

[8] LDC, "CALLHOME American English Lexicon," http://www.ldc.upenn.edu/.

[9] Y. Gong, "A method of joint compensation of additive and convolutive distortions for speaker-independent speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 13, no. 5, pp. 975–983, 2005.