MULTI-PASS PRONUNCIATION ADAPTATION

Nathan Bodenstab[1] *and Mark Fanty*[2]

[1]OGI at Oregon Health & Science University, USA
[2]Nuance Communications, USA
bodenstab@cslu.ogi.edu, mark.fanty@nuance.com

ABSTRACT

A mapping between words and pronunciations (potential phonetic realizations) is a key component of speech recognition systems. Traditionally, this has been encoded in a lexicon where each pronunciation is transcribed by a linguist or generated by a grapheme-to-phoneme algorithm. For large vocabulary recognition systems, this process is highly susceptible to errors. We present an off-line data driven algorithm to correct suboptimal pronunciations using transcribed utterances. Unlike previous data driven algorithms that struggle to balance acoustic representation and multi-speaker generalization, our multi-pass approach maximizes both criteria, instead of compromising between the two. We demonstrate on an automated name dialing task that our multi-pass algorithm achieves a 70% error rate reduction when compared to a baseline grapheme-to-phoneme generated lexicon.

Index Terms— pronunciation, speech, learning, adaptation

1. INTRODUCTION

Pronunciation adaptation has been found to significantly improve automatic speech recognition (ASR) performance in many applications, especially when applied to proper names [1,2,3]. In large vocabulary systems, tens of thousands of pronunciations must be generated to link each word with its possible phonetic realization(s). This is a difficult task for many reasons as speakers may have regional accents, vary in speaking style, speak in a non-native language, or exhibit speaking disabilities. Linguists must take these factors into account when constructing a lexicon for speech applications, and suboptimal entries are understandably prevalent, especially when audio samples are unavailable.

Two opposing objectives of pronunciation adaptation exist: to make a system more speaker-dependent, or more speaker-independent. Adaptation for personal dictation or other speaker-dependent tasks generate new pronunciations that purposely overfit the acoustic data, customizing the system to the individual. On the other hand, multi-speaker tasks, such as automated reservation booking, aim to make the system more speaker-independent, generalizing the acoustics of many audio samples into robust pronunciation candidates [4]. In this paper, we focus on the more difficult problem of adapting speaker-independent systems.

Prior work in this field dates back to the 1970s, and for many years pronunciation adaptation was carried out using phonological re-write rules fashioned by linguists. More recently, current data storage and processing capabilities allow us to analyze hundreds of hours of audio data, stimulating many data-driven adaptation techniques. Most algorithms produce new pronunciations via a two step process. First, a set of possible pronunciations are proposed with varying degrees of likelihood. Variations are often generated using rule-based methods or a phoneme recognizer. Once this set is constructed, audio samples are analyzed to determine which pronunciation variations should be added to the lexicon.

By definition, an adaptation algorithm must start from an initial location. In this paper, we refer to the initial pronunciation as the *canonical pronunciation*. Most adaptation algorithms assume this canonical pronunciation to be reasonably accurate, and employ forced alignment to rule-based variations of this initial candidate. Once this assumption is made, the algorithm must balance two (potentially) competing factors: acoustic representation and allowable deviation from the canonical source. Favoring the acoustic data can lead to overfitting and poor generalization, while favoring the canonical pronunciation limits the algorithm's ability to adapt. For words with poor canonical pronunciations, no optimal balance exists for a single-pass algorithm to produce new pronunciations that generalize well.

Our multi-pass algorithm sidesteps this problem by addressing each issue in turn. During the first pass, we analyze audio samples and derive frequent phonetic deviations from the canonical pronunciation. The second pass then constrains the set of possible pronunciation variations, and forces each audio sample to "choose" which pronunciation best represent its acoustics. Even when canonical pronunciations are inaccurate, our multi-pass algorithm produces pronunciations that both represent the acoustics and generalizes well to multiple speakers.

2. PRONUNCIATION OPTIMIZATION

Speaker-independent pronunciation adaptation seeks to find phoneme sequence(s) that express the acoustic data, yet still generalize to other speakers and speaking conditions. We will refer to these two factors as *acoustic representation* and *proper generalization*, the balance of each being vital to optimally adapting pronunciations. Formally, let X be an utterance (acoustic data), A be the canonical pronunciation, and B_i be the i^{th} proposed pronunciation. As in [3], the optimal pronunciation B^* can be found by solving

$$B^* = \arg\max_{B_i} P(B_i \mid A, X) \quad . \tag{1}$$

Using Bayes rule and simplifying where appropriate,

$$P(B_i \mid A, X) = \frac{P(A \mid X, B_i) P(X \mid B_i) P(B_i)}{P(X \mid A) P(A)} \qquad (2)$$

$$= \frac{P(X \mid B_i) P(B_i \mid A)}{P(X \mid A)} \ . \qquad (3)$$

Substituting Equation 3 into Equation 1 we get

$$B^* = \arg\max_{B_i} P(X \mid B_i) P(B_i \mid A) \ . \qquad (4)$$

As we see in Equation 4, the optimal phone sequence B^* decomposes into two independent components that parallel our prior optimization objectives: acoustic representation and proper generalization. Intuitively, the generalization term biases new pronunciations towards the canonical pronunciation, limiting the range of deviation we may incur by fitting the acoustics alone. Balancing these two elements is an important step to finding satisfactory pronunciations, but as we will see in Section 3.1, it does not solve the problem completely. The next two sections focus on these two components individually.

2.1. Acoustic Representation Model

The acoustic representation component of Equation 4 signifies the similarity between the phoneme sequence B_i and the instantiated phonemes embedded in the audio signal X. Our multi-pass algorithm does not compute this probability explicitly, but rather draws on a speech recognition engine to find the optimal pronunciation B^* when supplied with a finite state transducer of possible pronunciations, appropriately weighted with scores derived from the pronunciation distortion model.

Recognition is done with a speaker-independent system based on standard 3-state triphone HMMs, with a Genone-based state clustering mechanism [4]. The feature extraction front-end computes 27-dimensional mel-filterbank cepstral coefficients, with cepstral mean subtraction and standard noise reduction.

2.2. Pronunciation Distortion Model

To ensure new pronunciations do not overfit the acoustic data, we use the pronunciation distortion model to penalize possible deviations from the canonical phone sequence. For example, assume that the canonical phone sequence for the name 'Stephan' is A=`[s t E f @ n]` and we have competing pronunciations B_1=`[S s t E v A:]` and B_2=`[s t E f A: n]` (we use Computer Phonetic Alphabet symbols to represent phonemes). Although B_1 may represent the acoustics of one particular utterance very well (fitting 'S' to initial noise, replacing 'f' with 'v', dropping final nasal) it will most likely not generalize well to other utterances, so $P(B_1 \mid A)$ should be small. B_2, on the other hand, deviates by only one phoneme and is more likely to be robust across a wide range of speakers and speaking conditions. We expect $P(B_2 \mid A)$ to be large.

We compute the similarity between any two pronunciations, A and B, by first finding the optimal alignment between the respective phoneme sequences using a dynamic programming alignment algorithm [5]. We use the "empty" phoneme EPS to represent insertions and deletions. The optimal alignment between A and B_1 is seen in Table 1. This procedure guarantees that both phoneme sequences have an equal number of elements. Once alignment is complete, we assume independence between pairs of phonemes and calculate the probability

	Phone Sequence
Canonical	`EPS s t E f @ n`
Candidate	`S s t E v A: EPS`

Table 1. Dynamic programming alignment of two pronunciations for "Stephan" – [`s t E f @ n`*] and [*`S s t E v A:`*]. EPS is the empty phoneme.*

$$P(B \mid A) = \prod_{i=1}^{n} P(b_i \mid a_i) \ . \qquad (5)$$

The probabilities $P(p_i \mid p_j)$ for all p_i, p_j in the phoneme set (including EPS) correspond to a confusion matrix of phoneme substitutions. To estimate these probabilities, linguistic knowledge of phoneme similarity is necessary, as well as insertion and deletion frequency. Although it is possible to estimate these probabilities by hand, we choose to take a data-driven approach when sufficient data is available. Using a large lexicon with pronunciations transcribed by linguists, we extract each word with at least two pronunciations. We then align these pronunciations and tally the number of phoneme substitutions, insertions, and deletions. The counts from these linguist-certified phoneme transformations are used to derive appropriate phoneme substitution probabilities.

Multiplying simple pair-wise distortions in Equation 5 is an elegant way to obtain a similarity score between two phoneme sequences. Previous pronunciation adaptation algorithms [1,3] include additional context in their distortion model – e.g. $P(B \mid A) = \prod_{i=1}^{n} P(b_i \mid a_{i-1}, a_i, a_{i+1})$ – but we choose not to include this information for two reasons. First, accuracy results on pilot tests did not considerably improve with additional context; and second, this design decision allows straightforward pronunciation adaptation of languages with insufficient data to robustly estimate these context-rich probabilities. Simplifying the distortion model to a confusion matrix of phoneme substitution probabilities enables linguists to estimate these probabilities by hand in cases where data is limited.

3. MULTI-PASS ALGORITHM

As a running example, we will adapt the pronunciations of 25 utterances of "Stephan Granger". The canonical grapheme-to-phoneme pronunciations are presented in Table 2 and poorly represent the acoustic samples. The name is French, and properly pronounced `[s t E f A: n _ g r O: n dZ e/]`, but speakers of the 25 utterances have a mix of English or French as their native tongue. As a result, three pronunciations of 'Granger' are necessary to suitably cover speaker variation. These pronunciations can also be seen in Table 2.

The multi-pass algorithm processes all utterances with identical transcriptions as a single group (ie. uses the same

	Grapheme-to-phone	Linguist w/ Audio

Stephan	`s t E f @ n`
	`s t i: f @ n`
	`s t i: v @ n`
		`s t E f A: n`

Granger	`g r e/ n dZ r*`	`g r O: n dZ e/`
		`g r A: n dz i:`
		`g r & n dz r*`

Table 2. Prounciations for "Stephan Granger" generated by a grapheme-to-phoneme algorithm and a linguist with audio samples.

weighted finite state transducer) and all phonemes are optimized simultaneously. An outline of our proposed algorithm is as follows:

PASS 1a: Initialize phoneme lattice with canonical pronunciation(s). We represent a phoneme lattice of possible pronunciations using a weighted finite state transducer (WFST) with individual phonemes as input and the utterance transcription as output. If multiple canonical pronunciations are available, we construct a single WFST with each pronunciation in parallel. All negative log arc weights are set to zero.

PASS 1b: Augment WFST with phoneme transformations. We introduce all possible substitutions, insertions, and deletions into the WFST with probabilities from the pronunciation distortion model. Figure 1 shows a subset of the possible transformations to the original WFST of PASS 1a.

PASS 1c: **Find the best first-pass pronunciation for each utterance.** Using the recognition engine, find the optimal path through the WFST for each utterance and store the phoneme sequence.

PASS 2a: Extract phoneme transformations. Reassign the canonical pronunciation to the most frequent pronunciation in PASS 1c. Use the dynamic programming alignment algorithm to align the remaining pronunciations from PASS 1c to the new canonical pronunciation, and tally phoneme transformations. Table 3 shows PASS 2a results for "Granger". We discard the original canonical pronunciation if it was not returned by any utterance in PASS 1c.

PASS 2b: Construct second-pass un-weighted FST using the *n*-best transformations. Initialize a new FST with the new canonical pronunciations. Augment this FST with the *n*-best transformations extracted in PASS 2a, allowing at most 2^n possible pronunciations. Unlike PASS 1b, do not use probabilities from the pronunciation distortion model for two reasons. First, the PASS 2 FST uses only transformations that frequently occur, indicating that the transformation is worth learning. Second, we want to learn the true acoustics of the utterance set, so eliminating any bias favoring one phoneme over another is desirable.

PASS 2c: Find the best second-pass pronunciation for each utterance. Using the recognition engine, find the optimal path through the new FST for each utterance and store the phoneme sequence.

n-Best PASS 1 Transformations		PASS 3 Final Pronunciations	
Frq	Transformation	Frq	Pronunciation
10	**`e/`** `=> A:`	10	`g r O: n dZ e/`
9	**`e/`** `=> @:`	9	`g r A: n dZ i:`
2	**`e/`** `=> O:`	6	`g r O: n dZ r*`
5	**`r*`** `=> i:`		
4	**`r*`** `=> e/`		

Table 3. n-best transformations from PASS 1 recognition results of "Granger", and final pronunciations for "Granger" generated by the multi-pass algorithm.

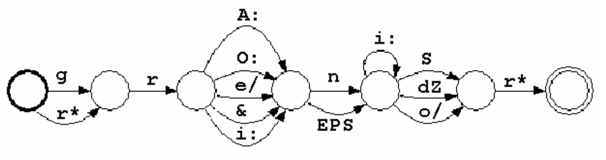

Figure 1. An example PASS 1 WFST for "Granger" with a subset of possible phoneme transformations.

PASS 3 (optional): Construct a third-pass un-weighted FST using the *m*-best pronunciations. A third pass is unnecessary for "stable" pronunciations, but for words such as "Granger" spoken with a variety of linguistic backgrounds, the results of PASS 2c may be uniform across possible pronunciations. It has been shown [4] that limiting the number of pronunciations added to a lexicon in large vocabulary system will increase performance, and for words with many unique acoustic realizations, a third pass is necessary to generalize pronunciations even further.

We select the *m*-best pronunciations from PASS 2c and build an un-weighted FST with only these *m* possibilities. Finding the optimal path through this new FST now forces each utterance to choose its closest pronunciation representative.

Update Lexicon. Add pronunciations from PASS 3 (or 2c) that have frequency counts above a set threshold. Depending on the application, these new pronunciations can supplement the original lexicon, or stand alone, pruning previous lexicon entries with no (or few) acoustic realizations. Table 3 shows the final pronunciations added by our multi-pass algorithm for the word "Granger".

3.1 Benefits of a Multi-Pass Algorithm

As is done in many pronunciation adaptation algorithms [3,6], we can incorporate an additional degree of freedom into Equation 4 by introducing the weighting parameter λ,

$$B^* = \arg\max_{B_i} \lambda \ln P(X \mid B_i) + (1-\lambda) \ln P(B_i \mid A) \quad . \qquad (6)$$

This additional parameter allows us to balance the contribution of acoustic representation and proper generalization, choosing to favor one over the other. Although beneficial, optimizing λ does not produce optimal pronunciations because we wish to satisfy both criteria, not compromise them. The next example makes this

λ =0.9		λ =0.1	
Frq	Pronunciation	Frq	Pronunciation
3	g r @ n dZ r*	18	**g r e/ n dZ r***
2	g r A: n dZ e/	3	g r O: n dZ r*
2	g @r A: n dZ i:	3	g r A: n dZ r*
1	r A: n dZ e/	1	g r A: n dZ e/
1	g r O: n Z r*		
1	g r @: n dZ r*		
1	g r @: Z e/		
1	g r ^ dZ r*		
1	g r A: n dZ i:		
	…		
0	**g r e/ n dZ r***		

Table 4. Pronunciations for 25 instances of "Granger" favoring the acoustics (0.9) and favoring the canonical pronunciation (in bold) (0.1).

point clear. Table 4 shows PASS 1 pronunciation results for "Granger" when λ is large, heavily favoring the acoustics of each utterance. Of the 25 utterances, 21 have unique pronunciations. Supplementing the lexicon with these pronunciations that have been overfit to the acoustic data causes greater recognition error in large vocabulary systems since the "pronunciation space" of each word expands and causes word confusion errors.

To inhibit pronunciations from fitting the acoustic too closely, we can shift λ closer to zero. Table 4 also shows pronunciation results of "Granger" with a lambda value of 0.1, strongly discouraging deviation from the canonical phone sequence. Although we know the canonical pronunciation to be incorrect, we see that a low lambda value does not allow adaptation to the correct pronunciations.

An assumption made by many pronunciation algorithms is that the canonical phone sequence requires only minor alterations (if any) to fit the desired utterances. But when significant deviation is necessary (e.g. Granger), no optimal lambda value exists that will produce the desired pronunciations. In these situations, our multi-pass algorithm has significant advantages by first favoring utterance acoustics and learning frequent phoneme transformations, then constraining the set of second-pass pronunciations to only possibilities that generalize well.

4. EXPERIMENTS

We evaluate the performance of our multi-pass algorithm on an automatic name dialing task. Name dialing applications benefit from pronunciation adaptation for many reasons, but primarily because pronunciations are difficult to produce, as linguists must infer information about the origin of the name and potential pronunciations of non-native speakers; rule-generated pronunciations are even worse.

The name dialing training and testing sets each consist of 3750 manually transcribed utterances, 25 instances of 150 unique names. The utterances of each set are unique and do not overlap. The weighting parameter λ is optimized on a validation set of different names to 0.75 and kept constant through all experiments. The final multi-pass lexicon contains only pronunciations which are preferred by at least 20% of the utterances in the training set. Accuracy is measured on correct recognition of an entire name, not per word. To better simulate a real automatic name dialing system, we increase the grammar to contain 10,000 names.

	Graph-to-phone	Linguist
Baseline	25.6	9.6
MP PASS 1	11.2	6.8
MP PASS 2	9.5	6.0
MP PASS 3	7.8	5.4

Table 5. Call-routing error rates on a name dialing task when canonical pronunciations are generated by a grapheme-to-phoneme algorithm or by a linguist.

We see in Table 5 that the multi-pass algorithm achieves a 70% reduction in name recognition error when tested against a rule-based grapheme-to-phoneme pronunciation generator, and 43% reduction in name recognition error when compared to a linguist-generated lexicon (note that the linguist had access only to the grapheme representation of words, and not the audio files, although this is a common practice when composing a lexicon). We also see in Table 5 that the baseline error rate of Linguist is higher than the best performance of MP PASS 3 using only rule-based pronunciations. Nevertheless, seeding the multi-pass algorithm with pronunciations from a linguist greatly increases the performance, achieving a 2.4% absolute reduction in name recognition error.

5. CONCLUSIONS

We have presented a new multi-pass approach to pronunciation adaptation that eliminates the prior need to compromise between acoustic representation and proper generalization. We have shown a 70% reduction in name recognition error rate when evaluated against a grapheme-to-phoneme generated lexicon, and a 43% reduction when initialized with canonical pronunciations from a linguist.

6. REFERENCES

[1] F. Bechet, R. de Mori, and G. Subsol, "Dynamic Generation of Proper Name Pronunciations for Directory Assistance," Proc. ICASSP '02, pp. I:745-748 vol. 1, 2002.

[2] I. Amdal, F. Korkmazskiy, and A. Surendran, "Joint Pronunciation Modelling of Non-Native Speakers Using Data-Driven Methods," Proc. ICSLP '00, pp. III:622-625, 2000.

[3] F. Beaufays, A. Sankar, S. Williams, and M. Weintraub, "Learning Name Pronunciations in Automatic Speech Recognition Systems," Proc. ICTAI '03, pp. 233-240, 2003.

[4] H. Strik, "Pronunciation Adaptation at the Lexical Level," Proc. ISCA ITRW Workshop Adaptation Methods for Speech Recognition, pp. 123-131, 2001.

[5] T. Vintsyuk, "Speech discrimination by dynamic programming," Kibernetika, pp. 4:81-88, 1968.

[6] J. Lucassen and R. Mercer, "An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms," Proc. ICASSP, pp. 42.5.1-42.5.4, 1984.